PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06K	A2	(11) International Publication Number: WO 97/04412 (43) International Publication Dat : 6 February 1997 (06.02.97)

(21) International Application Number: PCT/US96/11416

(22) International Filing Date: 8 July 1996 (08.07.96)

(30) Priority Data:
08/504,117 19 July 1995 (19.07.95) US

(71) Applicant: CABLE TELEVISION LABORATORIES, INC. [US/US]; 400 Centennial Parkway, Louisville, CO 80027 (US).

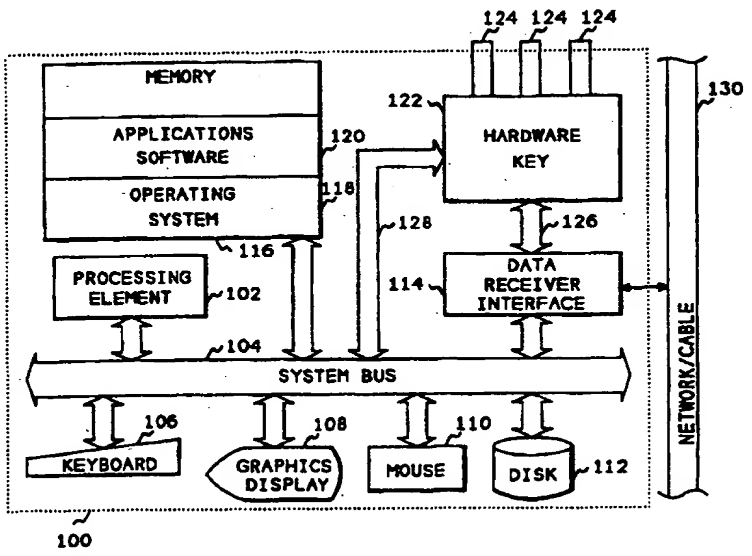
(72) Inventors: WILLIAMS, Thomas, H.; 6423 Fairways Drive, Longmont, CO 80503 (US). BAGGETT, Claude, T.; 962 Alder Way, Longmont, CO 80503 (US).

(74) Agent: YOUNG, James, R.; Suite 385, 12000 N. Washington Street, Denver, CO 80241 (US).

(81) Designated States: CA, GB.

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD FOR PROTECTING PUBLICLY DISTRIBUTED SOFTWARE



(57) Abstract

A system for protecting software from copying wherein the software to be protected is placed on the computer system in two parts. A first part (120) is stored in non-volatile storage, such as a hard disk or floppy disk within the computer system (100), and a second part is stored and executed in a "hardware key (122)", which is attached to the computer system (100). The second part is stored in volatile RAM (206) and will be erased when electrical power is removed from the hardware key (122), or when the software stops execution. This requires that the second part of the software be reloaded each time the hardware key (122) is powered up. Typically, the second part of the software will be loaded from a network (130), or from a cable network, thus reloading of the second part into the hardware key (122) is a trivial matter, so long as the user is an active subscriber to the network (130) or cable network.

METHOD FOR PROTECTING PUBLICLY DISTRIBUTED SOFTWARE

TECHNICAL FIELD

This invention relates to computer systems and more particularly to preventing use of copied software within such computer systems. Even more particularly, the invention relates to preventing use and copying of software that is publicly distributed, such as through a cable television connection or through a public network.

BACKGROUND OF THE INVENTION

Software copying is a problem that has plagued the software industry from the beginning of the industry. This problem is more severe for computer game software, because of the low cost of game software and because of the distribution methods used for game software. Even though business software is usually priced higher than game software and business software is often distributed in ways that better protect the software from copying, copying is still a significant problem for business software.

Some game software is now distributed via cable television. A special purpose computer system receives the software from a cable television game channel only into volatile storage, that is, storage not capable of storing the game software after electrical power is removed. The game software is captured by the special purpose computer through a data receiver as the software is transmitted over a cable television game channel, and the user plays the game though the special purpose computer system. The game remains within the memory of the special purpose computer system until electrical power is removed, thus the user can play the game until they turn off the special purpose computer. Since the special purpose computer system has no non-volatile memory that is used for storing the game, nor can the special purpose computer memory be accessed by a general purpose computer having non-volatile storage, the game cannot be saved or copied. This allows only users with access to the particular cable game channel(s) to play the game. If the user desires to play a new game, they wait until the new game is transmitted over the cable television game channel, capture the new game and play it as long as desired. If the user discontinues access to the cable television game channel, they can no longer access any games.

This method affords copy protection for the game as well as ensuring adequate revenue for the game supplier. However, it requires the special purpose computer in order to play the game.

The method will not work using a general purpose personal computer, such as an IBM compatible personal computer, or an Apple computer, because these types of computers do contain non-volatile storage in the form of floppy and hard disk drives. Using the non-volatile storage, the user could keep the game permanently, or give a copy to others, thus depriving the game supplier of revenue for the game.

Other methods have been tried to protect software. U.S. Patent 4,683,553 issued Jul. 28, 1987 to Mollier, entitled "Method and device for Protecting Software Delivered to a User by a Supplier" appears to use a card containing a validation code necessary to unlock a program and allow the program to execute. The card is needed each time the program is loaded into main memory for execution. The card contains only a validation code used to unscramble portions of the program. Once unscrambled, however, the software resides, unprotected, in the memory system of the computer, thus it is vulnerable to being copied and distributed to users that have not purchased the card.

U.S. Patent 4,819,267 issued Apr. 4, 1989 to Cargile, et al., entitled "Solid State Key for Controlling Access to Computer Systems and to Computer Software and/or for Secure Communications", describes an enabling device that attaches to a computer wherein the device computes a password, that is matched to a password computed by software in the computer. When the two passwords match, the enabling device returns authorization that allows the computer, or a specific piece of software in the computer, to operate.

U.S. Patent 5,222,133 issued Jun. 22, 1993 to Chou, et al., entitled "Method of Protecting computer Software from Unauthorized Execution Using Multiple Keys", uses two keys that have to match in some manner to allow the software to operate. The first key is stored in a hardware enabling device, plugged into an I/O port. The second key is typed in by a user. The software processes the two keys, and allows the software to continue if the keys match.

In both the Cargile and Chou, et al. systems, because the password is a single entity that can be captured as it is sent between these computer and the enabling device, the system is vulnerable to copying. That is, the password can be captured, and a device constructed to always return the "correct password" authorization.

U.S. Patent 5,212,729 issued May 18, 1993 to Schafer, entitled "Computer Data Security Device and Method" provides a hardware device, plugged into the parallel port of a computer, plus a method that encrypts and decrypts data stored on the hard disk. The system installs a patch into the boot block of the hard disk, and this boot patch interacts with stored code in the hardware device. Once the patch is installed, each time the computer boots up, the hardware device supplies a code to the boot program. If the correct code is provided to the boot program, access to the computer's disk is provided. The boot patch installs disk data encryption/decryption software into the computer's disk BIOS routine and thereafter is used at all times to transparently encode/decode partition data identifying the location of the data on the disk. Without the access code stored on the hardware device and proper password, the contents of the disk are unreadable. Both floppy and hard disks may be used with the device. The hardware device is in two parts -- a socket device that attaches to the parallel port, and a key device that plugs into the socket device. This device protects all software used in the computer system, thus it is not very effective for use by multiple software vendors, since they would all have to coordinate and use the same key for each customer. It is also very inconvenient for the user of the system, since anyone needing access to their data must have the password.

U.S. Patent 5,222,134 issued Jun. 22, 1993 to Waite, et al., entitled "Secure System for Activating Personal Computer Software at Remote Locations", describes a system wherein a registration program is run by the user after receiving the software. The registration program calls the vendor's registration computer which, after receiving required registration information, downloads an overlay portion of the software which allows the software to run thereafter. Once the overlay portion is downloaded, however, the software may be easily distributed without protection.

U.S. Patent 5,343,524 issued Aug. 30, 1994 to Mu, et al., entitled "Intelligent Security Device", shows the use of a hardware device, containing a microprocessor, attached to a serial port. This device contains some of the software needed to operate the program. The software in the hardware device is encrypted, so that, if copied, it will not function. The program running in the computer system to which the hardware device is attached sends a security code to the hardware device. The hardware device then decrypts its internal software needed to interpret the security code, and decrypts a security code within the hardware device and compares the two codes with the internal software. If everything matches, the hardware device returns ("downloads") some software to the computer system. This software is inserted into the program, which enables operation of the program. Mu, et al. also makes provision for two security codes to be used, one from the software creator, and one that is set by the user to allow only the specific user to operate the program. Mu, et al. asserts that the system is unbreakable, because the software and comparison security code stored in the hardware device are encrypted, and that any attempt to retrieve the software from the hardware device will destroy the encrypted software and security code. Mu, et al. discusses the use of a "local calculation step", and returning the correct answer in this step is necessary to pass the security check. However, once the hardware device returns the needed software to the computer system, the software is vulnerable to copying, since it is completely operable within the memory of the general purpose computer to which the hardware device is attached.

U.S. Patent 5,400,403 issued Mar. 21, 1995 to Fahn, et al., entitled "Abuse-Resistant Object Distribution System and Method", uses a pure software system which encrypts and then distributes the software via any method. Independently of the distribution of the software, an access number is distributed, such as through a retail store. When the user has both the software and the access number, the user's computer system contacts the distribution unit, over a secure distribution channel such as a telephone line, to get permission to decrypt. This permission is in the form of a second key, and when received, the two keys are used to decrypt the software for use. Once decrypted, however, the software is vulnerable to copying.

It is thus apparent that there is a need in the art for an improved method or apparatus which allows software to be distributed by several methods, and prevents the software from being copied after the software is installed into the memory of the computer system. The present invention meets these and other needs in the art.

DISCLOSURE OF THE INVENTION

It is an aspect of the invention to prevent unauthorized use and copying of publicly distributed software.

It is another aspect to distribute a part of the software using secure means for distribution.

Another aspect is to perform the secure part of the software using a processor in a hardware key that is separate from the processor in a general purpose computer to which the hardware key is attached.

The above and other aspects of the invention are accomplished in a system wherein the software being protected is divided into two parts. A first part is stored in non-volatile storage in the computer system, such as a hard disk or floppy disk, and a second part is always downloaded, through a computer network or cable network, into the memory of a "hardware key", which is attached to the computer system. The second part is erased each time electrical power is removed from the hardware key, since the hardware key ordinarily receives its electrical power from the same electrical power supply as the computer system. The second part may also be erased when the first part completes execution.

For the software to operate, the second part of the software must be reloaded each time the computer system, and thus the hardware key, is powered up. Typically, the second part of the software will be loaded from a computer network, or from a cable network or cable television data carrier, thus reloading of the second part into the hardware key is a simple and fast operation, so long as the user is an active subscriber to the computer network or cable network.

The hardware key executes some or all of the second part of the software when requested by the first part of the software. This may be done, for example, as a subroutine call, or by passing data to the second part and expecting different data in return, such as returning the value of a function or indexing a table to return a value from the table. Because the second part of the software executes in the hardware key, the request sent by the first part and the results returned by the second part may be very complex and thus appear essentially unpredictable to one attempting to copy the hardware key and the second part of the software. This prevents easy duplication of the hardware key functions. Since the two parts of the software execute in two different processors, and the memory containing the second part is not accessible from the computer system, the system prevents copying of usable software from the computer system memory. In addition, different applications or programs would ordinarily use different forms of returned data, because their functions would normally be different, thus the knowledge learned in breaking one program will be of little or no help in breaking a different program.

DESCRIPTION OF THE DRAWINGS

The above and other aspects, features, and advantages of the invention will be better understood by reading the following more particular description of the invention, presented in conjunction with the following drawings, wherein:

Fig. 1 shows a block diagram of a computer system that incorporates the hardware key and method of the present invention;

Fig. 2 shows a block diagram of the hardware key electronics of the present invention;

Fig. 3 shows a flowchart of the software used to capture the publicly distributed software;

Fig. 4 shows a flowchart of the method for performing the software;

Fig. 5 shows a flowchart of the method of capturing the software and decoding the secured parts of the software;

Fig. 6 shows a flowchart of the method of performing the part two subroutines; and

Fig. 7 shows a flowchart of the method of monitoring the execution of the software.

BEST MODE FOR CARRYING OUT THE INVENTION

The following description is of the best presently contemplated mode of carrying out the present invention. This description is not to be taken in a limiting sense but is made merely for the purpose of describing the general principles of the invention. The scope of the invention should be determined by referencing the appended claims.

Fig. 1 shows a block diagram of a computer system that incorporates the hardware key and method of the present invention. Referring now to Fig. 1, a computer system 100, which is typically a general purpose computer such as a personal computer, contains a processing element 102. The processing element 102 communicates to other elements of the computer system 100 over a system bus 104. A keyboard 106 and a mouse graphical input device 110 allow a user of the system 100 to input data, and a graphics display 108 allows software operating within the computer system 100 to output data or information to a user of the system. A disk 112 stores the first part of the software used with the present invention, and the disk 112 stores any data used in the present invention.

The computer system 100 also contains a memory 116 which stores an operating system 118 and application software 120 of the present invention. Although ordinarily present within general purpose computers, the operating system 118 is not necessary to the present invention.

A data receiver interface 114 is attached to the system bus 104 and also attached to a two-way computer network or one-way cable network 130. This interface is used to receive software from a supplier. When the interface 114 is attached to a cable network, the interface must contain a radio frequency data receiver, since data transmitted over cable television is transmitted at very high frequencies. When the interface 114 is connected to a network cable, the interface must receive data in the form sent over the network. For example, such a network might be an Ethernet, fiber distributed data interface (FDDI), token ring network, etc. or it may be a modulated carrier such as QPSK, 9-PRS, or 64QAM. The type of interface contained within the data receiver interface 114 is unimportant to the invention, so long as the interface is compatible with the signals transmitted on the cable 130.

Also attached to the system bus 104 is a hardware key 122. A more detailed block diagram of this device is shown below with respect to Fig. 2. The hardware key 122 is also connected directly to the data receiver interface 114 through an interface 126. This allows the hardware key 122 to receive data directly from the network 130 without having this data pass through the system bus or any other portion of the computer system 100. The hardware key 122 receives and decodes the secure subroutines used as part of the present invention.

Optionally attached to the hardware key 122 is one or more ISO-7816 compatible cards 124. These ISO-7816 compatible cards conform to the ISO-7816 standard which has the same connector pattern as the national renewable security standard (NRSS) card. When present, the ISO-7816 compatible card or NRSS card provides additional security or renewable security in conjunction with the hardware as part of the present invention.

Fig. 2 shows a block diagram of the electronics for the hardware key 122 shown above in Fig. 1. Referring now to Fig. 2, the electronics contained in the hardware key comprise a small computer system. Thus, the hardware key contains a processor 202 which communicates to other elements of the hardware key over a hardware key bus 204. A memory 206 is used to contain the secure subroutines of the present invention. A system bus interface 210 is used to interface the hardware key to the system bus 104 shown above with respect to Fig. 1. A data receiver interface 212 is used to interface the hardware key to the data receiver interface 114 shown above with respect to Fig. 1.

Interface 208 is used to connect the hardware key 122 with one or more ISO-7816 compatible cards 124 as described above with respect to Fig. 1.

The present invention is designed to protect software from copying by having the software delivered in two parts. The first part, which typically comprises the majority of the software, will be delivered through normal distribution channels or alternatively may be broadcast over cable television and received through the data receiver interface 114 and stored on the disk 112 (Fig. 1). Alternatively, the first part of the software can be distributed in any manner, even publicly accessible manners, such as through bulletin boards, or commercial software providers.

In order to use the software, the user must purchase a hardware key, containing a unique address, from a retail store or mail order, and the user must subscribe to a cable network, or other network service, used to distribute the software. The unique address could be in any form, such as a series of letters and numbers that are stored in the hardware key, or they may be derived from the ISO-7816 compatible card(s) plugged into the hardware key 122. The provider of the software insures payment by controlling access to the unique address or the ISO-7816 compatible cards or the network.

Once the first part of the software is stored on the disk 112, and the unique address is provided to the hardware key 122, the hardware key then captures the second part of the software from the network 130 through the data receiver interface 114. Because the second part of the software is captured directly and placed in the memory 206 of the hardware key without ever being transferred into the memory 116 or the hard disk 112 of the computer system 100, the software comprising the second part exists only within the memory 206 of the hardware key 122. Because the memory 206 is not directly addressable from the computer system 100, the second part of the software contained in the memory 206 cannot be copied into the computer system 100, and therefore cannot be copied along with the first part of the software in order to make the entire software available. Ideally, the data destined for memory 206 is encrypted while on the network and is decrypted when received by the hardware key before being stored in the memory 206.

The first and second parts of the software cooperate to accomplish the intended function of the software. This cooperation is accomplished by the first part performing the majority of functions that accomplish the intended operation of the software, however, on occasion the first part will call the second part, for example in the manner of calling a subroutine, to receive data or results necessary to complete the functions of the software. When the second part is called, it operates entirely within the processor contained within the hardware key 122 to produce the result needed by the first part and to send this result back over the interface 128 to the first part of software, that is operating within the memory 116 of the computer system 100. Because the second part of software comprises one or more subroutines, these subroutines would typically perform at least moderately complex functions, so that the data returned over the bus 128 cannot easily be cloned or mimicked by one seeking to pirate the software.

Thus, the software protection is accomplished by dividing the software into two parts, having one of those parts exist entirely within the hardware key 122, and having the functions performed by this second part be too complex to be easily duplicated.

Because the hardware key is a small computer system, it will have control software that is used to control communication between the hardware key and the processor 102, and between the hardware key and the data receiver interface 114. Typically, this control software will be contained in a Read Only Memory (ROM) device within memory 206. Alternatively, the control software may be contained within a volatile part of memory 206 that is powered by a battery 214. Thus, if the user attempts to tamper with the hardware key by disassembling it, the battery will be removed and the control software erased.

Figs. 3-7 show flowcharts of the functioning of the software. Fig. 3 shows a flowchart of software operating within the computer system 100 that is used to capture the publicly distributed first part of the application software. Referring now to Fig. 3, after being invoked through a command supplied by the user, control enters Fig. 3 at block 302 which determines whether the data receiver interface 114 is connected to a 1-way or to a 2-way network. If connected to a 2-way network, such as a computer network, block 302 transfers to block 304 which sends a request over the network to have a server device send the software to the computer system 100.

If the connection is to a 1-way network, such as a cable network, block 302 connects to block 306 which monitors the network through the data receiver interface 114 to determine when the first part of the software is being transmitted. Typically, the first part of the software would be transmitted over the cable network at infrequent intervals, perhaps only once every ten minutes. Thus, a user wishing to first utilize the software must wait perhaps as long as ten minutes for the software to arrive. However, since this software is subsequently stored on the disk 112, this wait only occurs the very first time the user uses the application software.

After sending the request to the network server, or after waiting until the software is transmitted over the network, control goes to block 308 which receives the software through the data receiver interface 114. Block 310 then stores the first part of the software onto the disk 112 for subsequent use.

Alternatively, the flowchart of Fig. 3 can be eliminated entirely by obtaining the first part of the software through retail outlets or other public distribution systems. For example, the first part of the software might be freely distributed on a CD-ROM along with many other programs, since the first part is inoperable without the hardware key obtaining the second part.

Fig. 4 shows a flowchart of the method for running the application software. Referring now to Fig. 4, the software of Fig. 4 is executed by the user entering a command through the keyboard 106 or by clicking an icon using the mouse 110. After entry, block 402 loads the application software from the disk 112 where it was stored using the process of Fig. 3. Alternatively, the first part could be loaded directly from the network at this time.

Once the software is loaded, block 404 sends a capture request for the secure subroutines to the hardware key 122. Block 406 waits until the secure subroutines have been received, and optionally decoded or decrypted, and then block 408 transfers control to the application software. Once the application software starts, block 410 performs all the functions of the software that are resident in the first part, and then block 412 calls any secure subroutines that have been received by the hardware key.

Each set of application software may be different in how it uses the secure subroutines. The first part of the software may contain all the software with the exception of a single secure subroutine, or many secure subroutines may be used from various locations within the first part of the software. Typically, the software would be designed such that the second part, containing the secure subroutines, performs moderately complex functions which are not easy to duplicate. This would prevent the substitution of unauthorized hardware in place of the hardware key 122.

Block 414 then determines whether the software is complete and if not, returns to block 410. The loop comprising blocks 410, 412 and 414 continue so long as the user is using the application software. Once the user terminates the use of the application software, block 414 transfers to block

416 which calls a terminate secure subroutine within the hardware key 122 to terminate use of the program and erase the second part of the software contained in the memory 206 of the hardware key 122.

Alternatively, block 416 can be eliminated such that the secure subroutines remain in the hardware key 122 until electrical power is removed from the hardware key 122, which would typically be done by removing electrical power from the entire computer system 100.

Fig. 5 shows a flowchart of the method of capturing the software and decoding the secured parts of the software. This method operates in the hardware key 122. The method of Fig. 5 is called by block 308 of Fig. 3 and also by block 404 of Fig. 4. Referring now to Fig. 5, after entry, block 502 gets the next software data frame from the data receiver interface 114. This may require a waiting period, since in the case of a cable network, the parts for many different application software programs will be transmitted one after another. Typically the unsecure, first parts, of various programs would be transmitted infrequently, however, the secure, part two, software for any individual program would typically be transmitted every few seconds.

Once the software data frame is received, control goes to block 504 which determines whether the frame received is for the requested software and if not, block 504 transfers back to block 502 to get the next software data frame. If the software data frame does contain the requested software, block 504 transfers to block 506 which determines the data type. If the data frame contains unsecured data, for part one of the software, block 506 transfers to block 514 which transfers the data block to the local computer 100 over the interface 128. This section of the software will not need to be performed if part one of the software is already stored within the computer system.

If the data frame received is secured data, for part two of the software, block 506 transfers to block 508 which determines whether the access to this secure data frame is authorized. The access will be authorized if the user has supplied the correct access code for this particular application software program. If the access is authorized, which may be automatic through access to the cable channel, block 508 transfers to block 510 which decodes the data frame and then block 512 stores the data in the hardware key memory 206. The software decoding of block 510 may include decrypting the software.

Alternatively, the decision of block 506 may be based solely on whether part one or part two of the software is being received. Although part two of the software would normally be secured, there is no requirement that part two be secured while being sent to the computer system.

After storing the data in the hardware key memory, or if the access was not authorized, or after sending the data to the local computer in the case of a part one data frame, control goes to block 516 which determines whether all data frames have been received and if not, block 516 transfers back to block 502 to get the next software data frame. After all data frames have been received, block 516 transfers to block 518 which sends a completed indicator to the local computer to indicate that the software has all been received.

Fig. 6 shows a flowchart of the method of performing the secured subroutines. Referring now to Fig. 6, when the secured subroutine is called by sending a message over the interface 128 to the hardware key 122, control enters at block 602 which performs the desired function as requested by the first part of the software. Block 604 then returns the result to the first part of the software and block 606 resets an optional timer, which may be used to determine whether the first part of the software is continuing to operate.

Fig. 7 shows a flowchart of the method of monitoring the execution of the software within the hardware key 122. As discussed above with respect to Fig. 6, each time a secured subroutine is called, a timer may be reset. Expiration of the timer indicates that the first part of the software is no longer operating within the memory 116 of the computer system 100. In this case, the hardware key erases the secured subroutines from the memory 206.

Referring now to Fig. 7, Fig. 7 is entered each time a clock tick occurs within the hardware key internal clock. After entry, block 702 increments a time value and block 704 determines whether the time has expired. If the time has not expired, block 704 returns, doing nothing else. If the time has expired, however, block 704 transfers to block 706 which erases the secured subroutines from the memory 206 of the hardware key 122.

Other timer methods could be incremented such that the timing is done in hardware rather than software and the timer interrupt would then occur only after the timer had expired.

Having thus described a presently preferred embodiment of the present invention, it will be understood by those skilled in the art that many changes in construction and circuitry and widely differing embodiments and applications of the invention will suggest themselves without departing from the scope of the present invention as defined in the claims. The disclosures and the description herein are intended to be illustrative and are not in any sense limiting of the invention, defined in scope by the following claims.

CLAIMS

What is claimed is:

1. A system for protecting computer software (120) from unauthorized execution within a computer system (100), the system comprising:

a memory (116) within the computer system for containing a first part of the computer software;

a hardware key (122) attached to the computer system, the hardware key (122) for containing and executing a second part of the computer software;

a processor (102) for executing the first part (120) of the computer software and for communicating to the hardware key (122) to cause execution of the second part of the computer software within the hardware key (122), when the execution of the second part of the computer software is requested by the first part of the computer software.

2. The system of claim 1 further comprising:

a data receiver (114) attached to the computer system and further attached to a transmission facility (130), the data receiver for receiving the second part of the computer software and storing the second part of the computer software into the hardware key.

3. The system of claim 2 further comprising a connection (126) between the hardware key (122) and the data receiver (114), and wherein the hardware key (122) will only receive the second part of the computer software through the connection (126).

4. The system of claim 3 wherein the first part (120) of the computer software causes the data receiver (114) to receive the second part of the computer software.

5. The system of claim 1 wherein the second part of the computer software is decoded by the hardware key (122) before execution of the second part of the computer software.

6. The system of claim 5 wherein the hardware key (122) further comprises a decoding device connected to the hardware key, wherein the decoding device decodes the second part of the computer software before execution of second part of the computer software in the hardware key.

7. The system of claim 1 wherein memory (206) within the hardware key (122) comprises only volatile memory for storing the second part of the computer software, wherein contents of the volatile memory are erased upon removing electrical power from the hardware key.

8. The system of claim 1 wherein the second part of the computer software comprises machine instructions for erasing the second part of the computer software when the hardware key (122) has not received communications from the computer system (100) within a predetermined amount of time.

9. The system of claim 1 wherein the hardware key further comprises a battery powered portion of volatile memory wherein the battery powered portion of memory contains control software.

10. A method for protecting computer software from unauthorized execution within a computer system, the method comprising the steps of:

(a) dividing the computer software into two parts;

(b) storing a first part (120) of the computer software into a memory (116) within the computer system;

(c) storing a second part of the computer software into a hardware key (122) attached to the computer system;

(d) executing the first part (120) of the computer software in a processor (102) contained in the computer system, wherein the first part of the computer software communicates to the hardware key (122) to cause execution of the second part of the computer software within the hardware key (122) to provide results needed for operation of the first part (120) of the computer software.

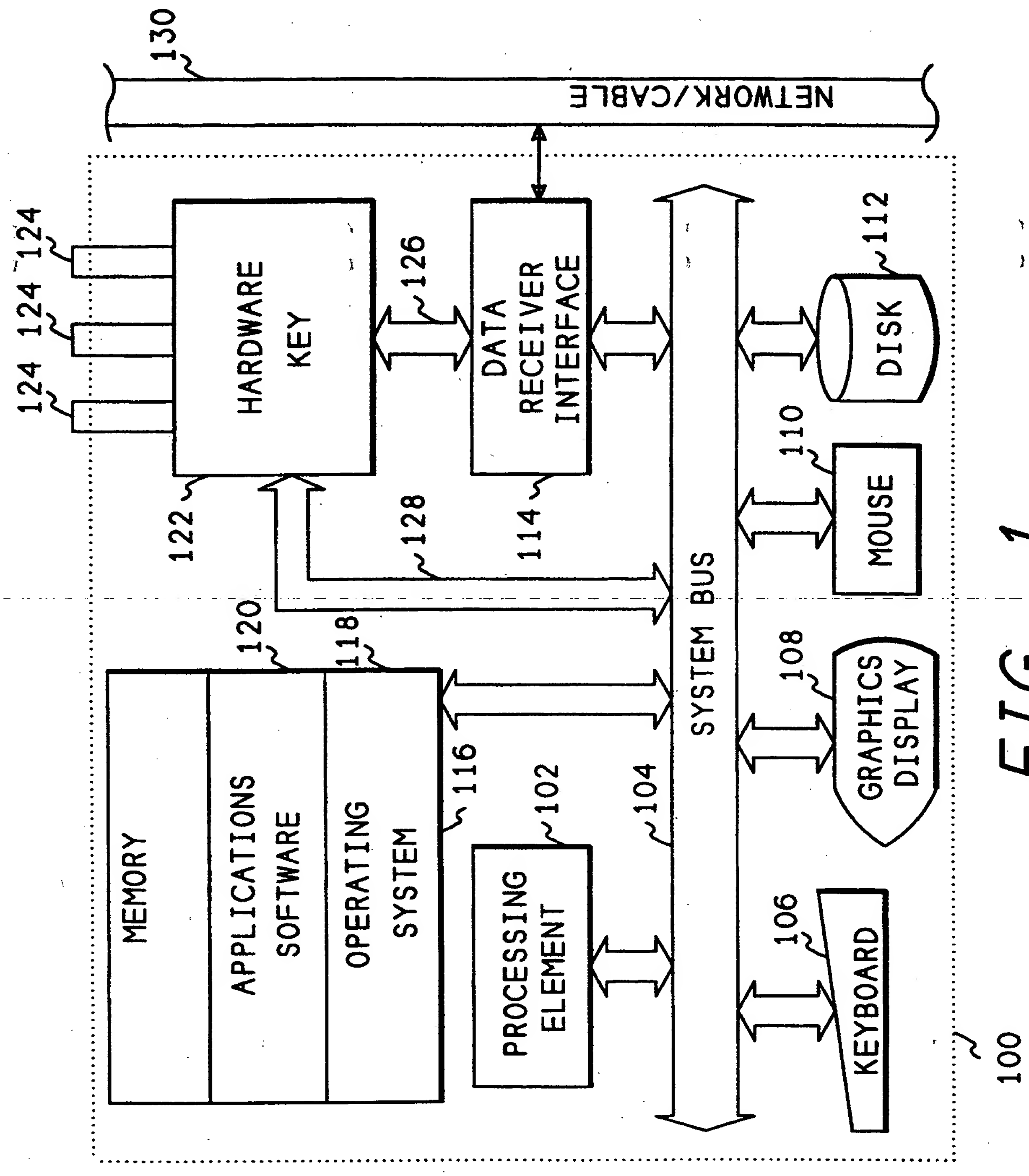
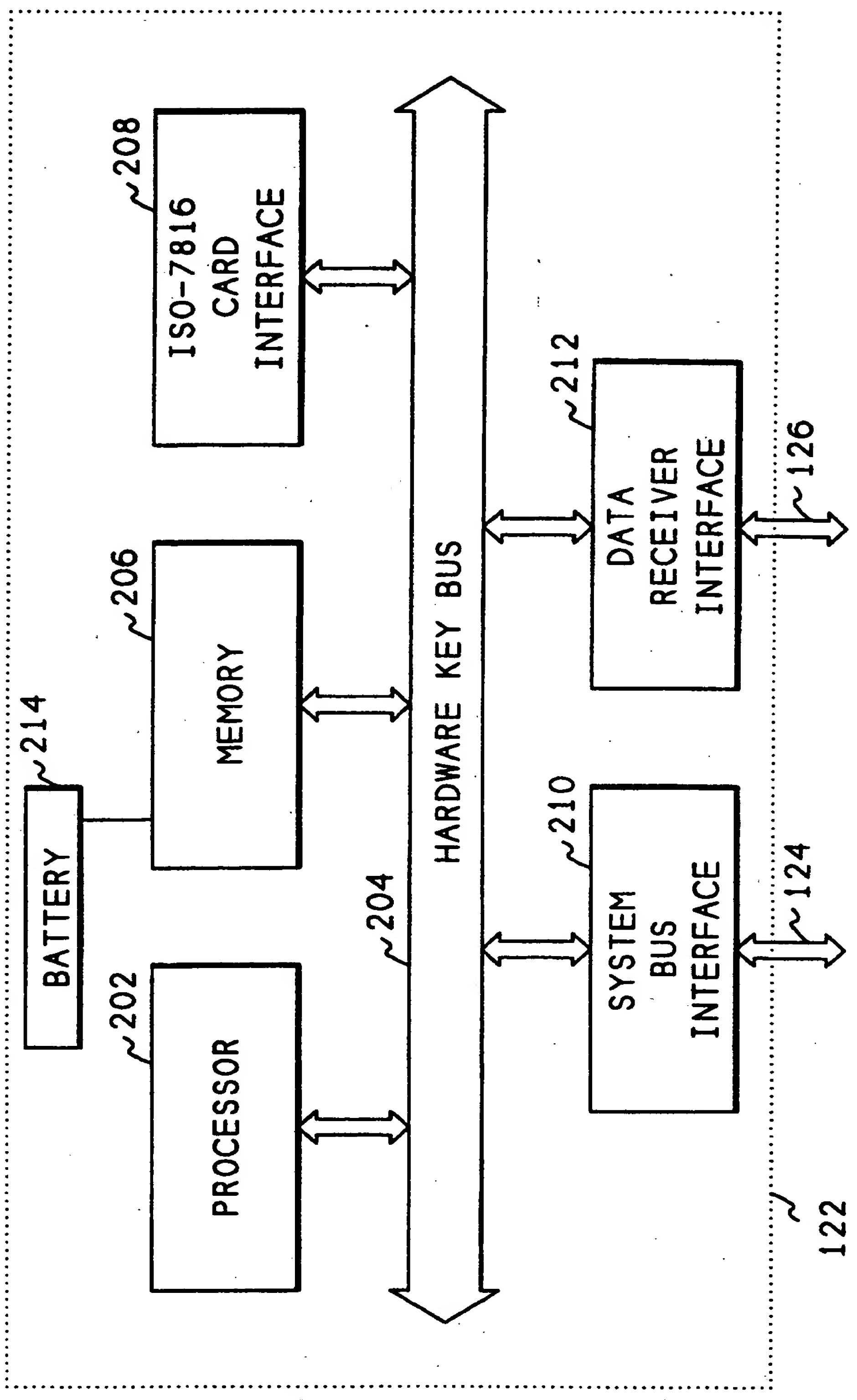

130
NETWORK/CABLE
124
124
124
HARDWARE KEY
126
DATA RECEIVER INTERFACE
122
128
114
112
DISK
110
MOUSE
SYSTEM BUS
120
118
MEMORY
APPLICATIONS SOFTWARE
OPERATING SYSTEM
116
102
104
PROCESSING ELEMENT
108
GRAPHICS DISPLAY
106
KEYBOARD
100

FIG. 1



ISO-7816 CARD INTERFACE
208
MEMORY
206
BATTERY
214
PROCESSOR
202
HARDWARE KEY BUS
204
DATA RECEIVER INTERFACE
212
126
SYSTEM BUS INTERFACE
210
124
122

FIG. 2

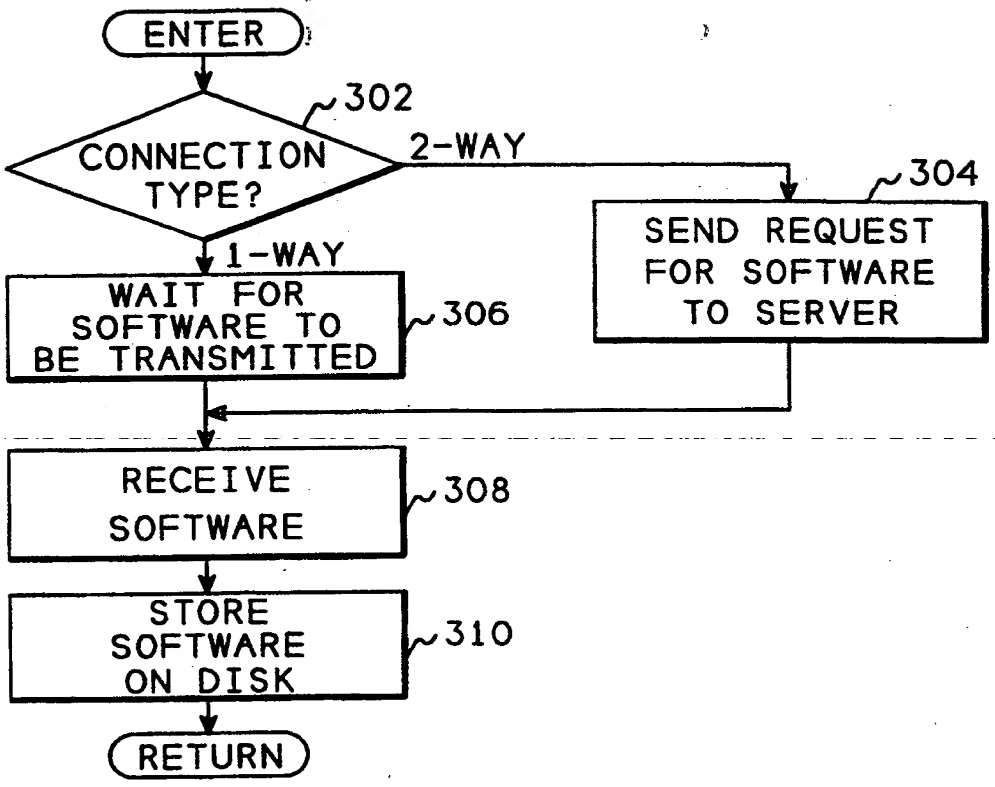

ENTER
CONNECTION TYPE?
302
2-WAY
1-WAY
SEND REQUEST FOR SOFTWARE TO SERVER
304
WAIT FOR SOFTWARE TO BE TRANSMITTED
306
RECEIVE SOFTWARE
308
STORE SOFTWARE ON DISK
310
RETURN

FIG. 3

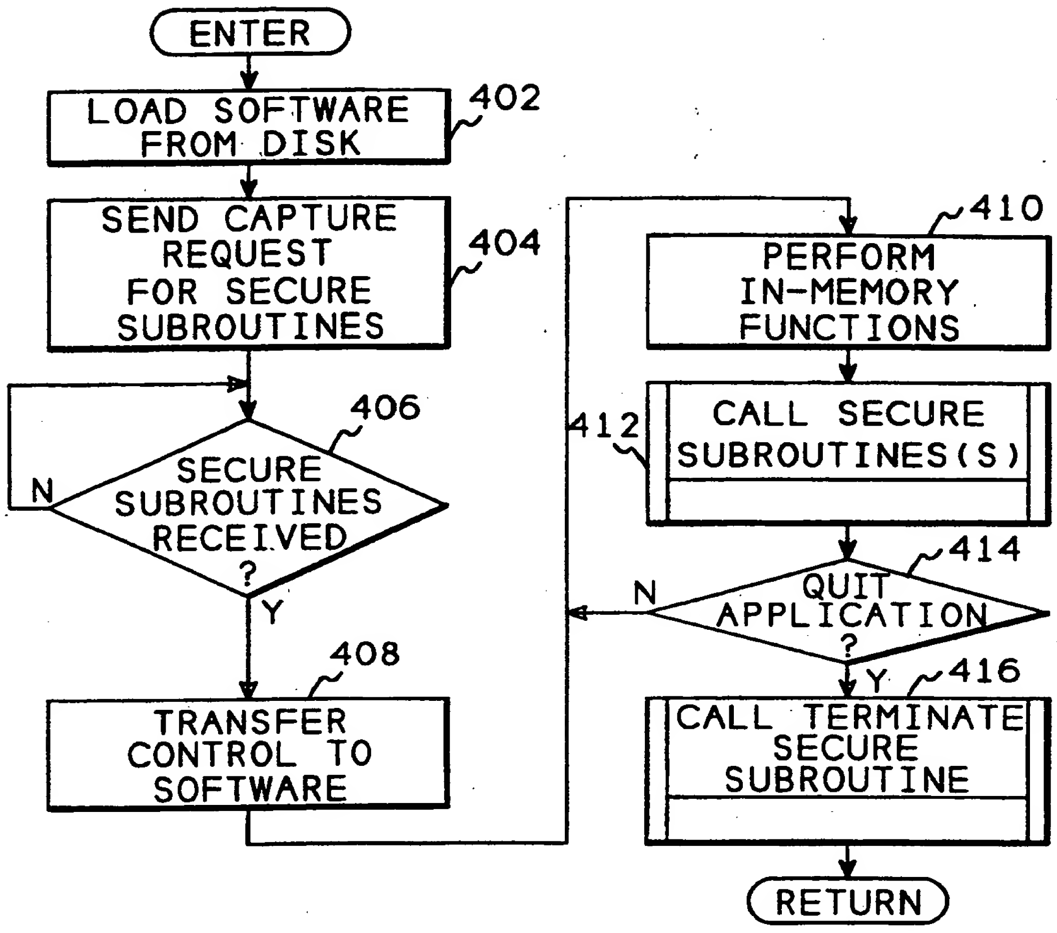

ENTER
LOAD SOFTWARE FROM DISK
402
SEND CAPTURE REQUEST FOR SECURE SUBROUTINES
404
SECURE SUBROUTINES RECEIVED ?
406
N
Y
TRANSFER CONTROL TO SOFTWARE
408
PERFORM IN-MEMORY FUNCTIONS
410
CALL SECURE SUBROUTINES(S)
412
QUIT APPLICATION ?
414
N
Y
CALL TERMINATE SECURE SUBROUTINE
416
RETURN

FIG. 4

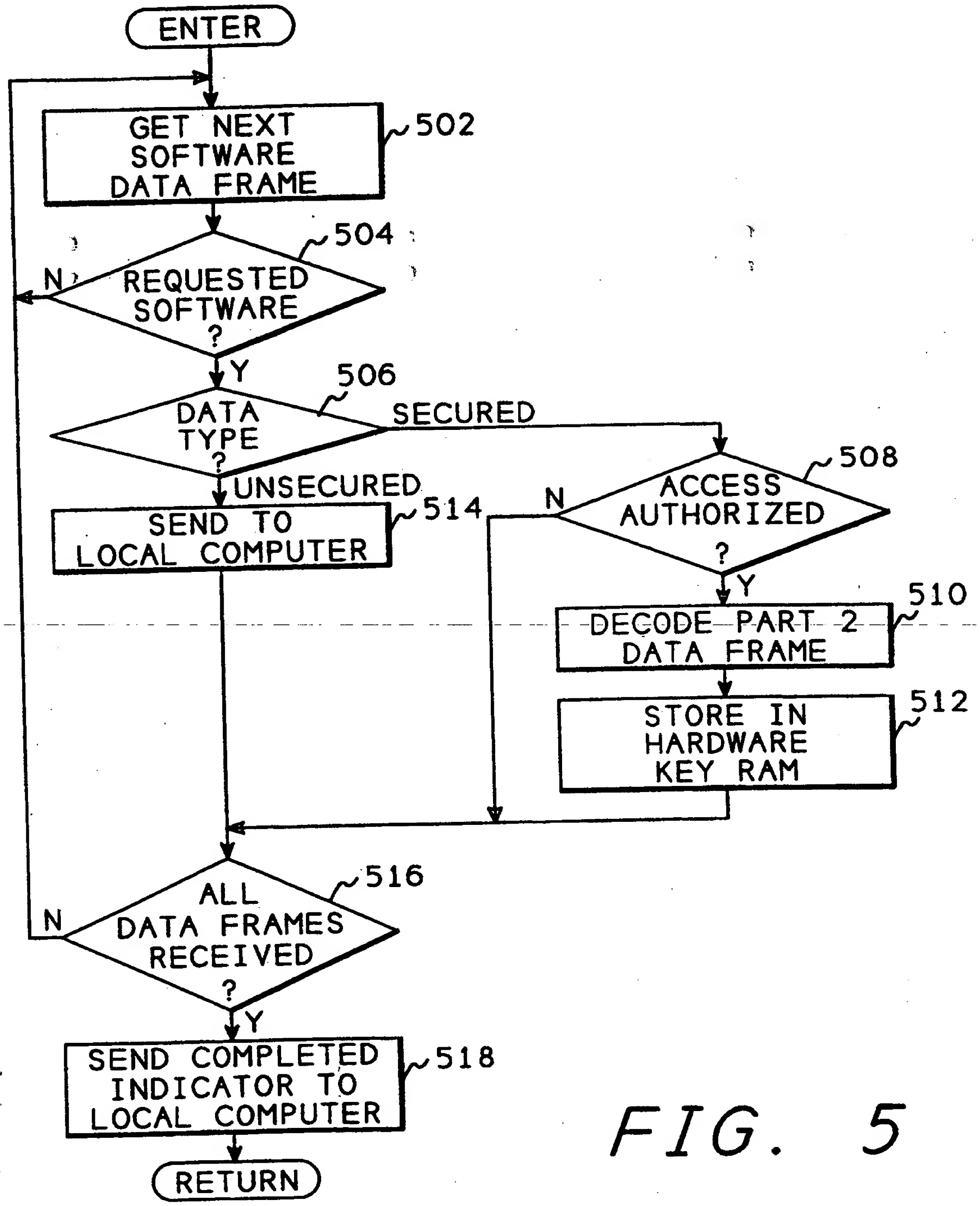

ENTER
GET NEXT SOFTWARE DATA FRAME
502
REQUESTED SOFTWARE ?
504
N
Y
DATA TYPE ?
506
SECURED
UNSECURED
ACCESS AUTHORIZED ?
508
N
Y
SEND TO LOCAL COMPUTER
514
DECODE PART 2 DATA FRAME
510
STORE IN HARDWARE KEY RAM
512
ALL DATA FRAMES RECEIVED ?
516
N
Y
SEND COMPLETED INDICATOR TO LOCAL COMPUTER
518
RETURN

FIG. 5

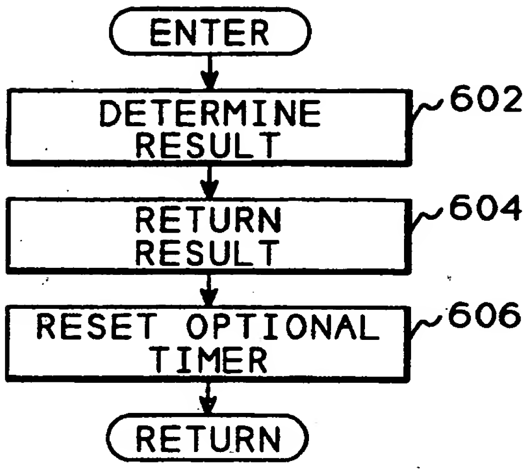

ENTER
DETERMINE RESULT
602
RETURN RESULT
604
RESET OPTIONAL TIMER
606
RETURN

FIG. 6

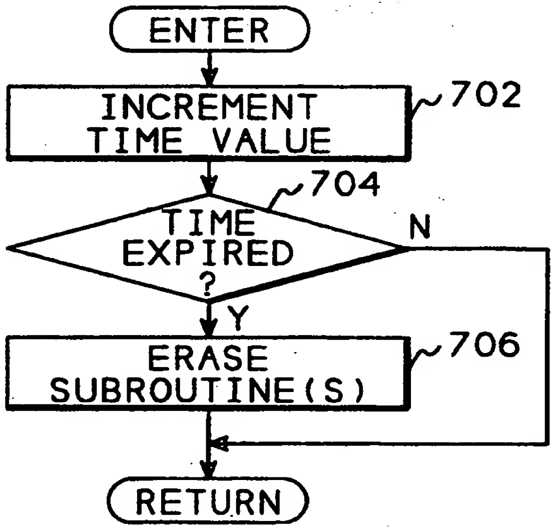

ENTER
INCREMENT TIME VALUE
702
704
TIME EXPIRED ?
N
Y
ERASE SUBROUTINE(S)
706
RETURN

FIG. 7